Brock University

Department of Computer Science

Safe Control Architectures for Mobile Robots Interacting with Humans

Jerzy A. Barchanski

Technical Report # CS-07-07
November 2007

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

Safe Control Architectures for Mobile Robots Interacting with Humans

Jerzy A. Barchanski
Dept. of Computer Science
Brock University
St. Catharines, Ontario
Canada
Jerzy.Barchanski@BrockU.ca

Abstract - **We review at first the principal concepts of system safety. Then we analyze hazards of robot – human interactions – human injury and unsafe communication. We focus on hazard of collision with a human and an appropriate robot behavior mitigating this hazard. After reviewing the principal classes of robot control architectures, we evaluate their effectiveness in obstacle avoidance.**

Index terms – safety, robot control architecture, human-robot interaction

I. INTRODUCTION

When robots interacts with humans, safety of the interaction becomes of paramount importance. It must be built into a robotic system from the start; it is difficult, if not impossible, to add it in an adequate and cost-effective manner later on.

The focus of this paper is on safety of control architectures for mobile robots interacting with humans. The term control architecture is usually used to mean a set of functional components of a robot control system and the principles of their interaction [1]. In addition to providing a structure it imposes constraints on the way a robot control problem can be solved. The components of an architecture are implemented mostly in software. This software allows unprecedented complexity of robotic systems, which goes beyond the ability of current engineering techniques for assuring acceptable risk. Selection of a right control architecture is crucial for robot safety. The present practice in mobile robotics does not follow any systematic approach to evaluation of their safety, such as e.g. Safeware Engineering [2]. As a result, the robots may be robust beyond a need (or not) but their level of safety is actually unknown. An example of such a case is [3].

In our previous research we have applied an element of Safeware Engineeing, so called Intent Specification [4] to development of safe robot control software [5].

In this paper we present our approach to safety analysis of robot control architectures based on system theory [6]. As far as we know this is the first attempt to evaluate robot control architectures from the viewpoint of safety.

The next section presents the principal concepts of system safety. In the following section we consider as an example hazard analysis of a robot built according to the Laws of Robotics, proposed by Isaac Asimow [7]. Then, we review the principles of the most important robot control architectures and evaluate their safety.

II. PRINCIPAL CONCEPTS OF SYSTEM SAFETY

Safety refers to the ability of a system to operate without causing an accident or an unacceptable loss [2]. An accident is an undesired and unplanned (not necessarily unexpected) event that results in (at least) a specified level of loss. To prevent accidents, something must be known about their precursors, and these precursors must be under control of the system designer. To satisfy these requirements, system safety uses the concept of a hazard. There are many definitions of hazards. We will define a hazard as a state or set of conditions of a system that, together with other conditions in the environment of the system may lead to an accident (or loss). We define therefore a hazard with respect to the environment of the robot.

A hazard has two important characteristics: severity and likelihood of occurrence.

Hazard severity is defined as the worst possible accident that could result from the hazard given the environment in its most unfavourable state.

The hazard likelihood of occurrence can be specified either qualitatively or quantitatively. Unfortunately, when the system is being designed

and hazards are being analysed and ranked as to which should be eliminated first, the information needed to evaluate the likelihood accurately is almost never available. It means, the best what can be done is to evaluate the likelihood qualitatively.

III. SYSTEM APPROACH TO HAZARD ANALYSIS

Traditional hazard analyses consist of identifying events that could lead the system to a hazardous state. These events are usually organized into causal chains or trees. Popular event-based hazard analysis techniques include Fault Tree Analysis (FTA) and Failure Modes and Effects Criticality Analysis (FMECA) [2]. Because of their reliance on discrete failure events, neither of these techniques adequately handles system accidents that result from dysfunctional interactions between system components.

We use for safety analysis of robot control architectures system-theoretical approach, similar to [8], which allows more complex relationships between events to be considered and also provides a way to look more deeply at why the events occurred. Whereas industrial safety models focus on unsafe acts or conditions and reliability engineering emphasizes failure events, a systems approach to safety takes a broader view by focusing on what was wrong with the system's design or operations that allowed the accident to take place.

The systems approach focuses on systems taken as a whole, not on the parts examined separately. It assumes that some properties of systems can only be treated adequately in their entirety, taking into account all facets relating the social to technical aspects. These system properties derive from the relationships between the parts of systems: how the parts interact and fit together. Thus the system approach concentrates on the analysis and design of the system as a whole as distinct from the components or the parts. While components may be constructed in a modular fashion, the original analysis and decomposition must be performed top down.

Safety is an emergent property of systems. Determining whether a plant is acceptable safe is not possible by examining a single valve in the plant. In fact, statements about the "safety of the valve" without information about the context in which the valve is used, are meaningless. In a system-theoretic view of safety the emergent safety properties are controlled or enforced by a set of safety constraints related to the behavior of the system components. Safety constraints specify those relationships among system variables or components that constitute the non-hazardous or safe system states. Accidents result from interactions among system components that violate these constraints – in other words, from a lack of appropriate constraints on system behavior.

IV. HAZARD ANALYSIS OF AN EXEMPLARY CASE

As an example we will consider a mobile robot acting in a world where the three Laws of Robotics [7] should be satisfied. We will be concerned with the hazards resulting from the first two Laws, which are relevant for human-robot interactions :

First Law: "A robot may not injure a human being, or through inaction, may not allow a human being to come to harm" "

From the first half of the Law:

"A robot may not injure a human being" – the hazard is a harmful physical contact with a human. It can be mitigated by a safety constraint inherent in the robot behavior **avoid _obstacles.**

The second part of the law :

"A robot through inaction, may not allow a human being to come to harm" envisioned by Asimov as a kind of action requiring sacrifice of robot existence to protect its master, is left for future generations of roboticists. At present it may be implemented at most aposteriori by search and rescue robots.

The second law: "A robot must obey the orders, given to it by human beings except where such orders would conflict with the First Law."

This law has to do with tasks commanded by a human, like: **move-to-goal**, **grasp**, **collect-pucks**, etc. The robot must be able to communicate with a human operator to execute his orders. The examples of hazards violating this law are communication security hazards – e.g. receiving from an illegitimate operator false commands or requests which may cause wrong actions. To eliminate such hazards it must be possible to authenticate the human operator.

Most conventional authentication protocols (e.g. Kerberos) require usage of an online server. This is out of the question here. Another widely used authentication protocol which seems to be more suitable is one based on public key cryptography. It was in fact proposed for authentication of cooperating autonomous digger and dumper truck [9]. However the problem of online access to the server appears here as well.

What we need is to be able to create a *secure transient association.* As well as being *secure*, the association must also be *transient*. When an operator changes, the robot has to obey the new operator.

A solution for this dilemma is to use a metaphor of a duckling emerging from its egg – it will recognize

as its mother the first moving object it sees that makes a sound, regardless of what it looks like: this phenomenon is called **imprinting.** Similarly, our robot will recognize as its owner the first entity that gives it a secret key. As soon as this imprinting key is received, the robot is no longer a newborn and will stay faithful to its owner for the rest of its life. The association is secure.
To make the association transient the robot must be designed so it can die (lose its memory) and resurrect (be imprinted again).
This security policy model, called "Resurrecting Duckling" [10] can be used not only in the communications between a human and a robot but between two robots as well, enabling robot-to-robot secure interaction
We will focus this paper on the hazard involving collision with an obstacle, whether it is a human or not. To deal with this hazard, a robot should be able first to detect the obstacle, then recognize it and finally execute the avoidance manoeuvre.
Obstacle recognition is useful only if the robot is supposed to cooperate with a human or another robot. Recognition of a human is quite a difficult task. It is necessary to use for this some special sensors (e.g. a suite of vision and thermal sensors).
Recognition of another robot may be easier as it is possible to give the robot a special appearance (e.g. color)
To reduce the risk of collision, robots must have some kind of a behavior to avoid obstacles. This behavior should be active all the time, concurrently with any other behavior active at the moment. We will discuss in the following how efficient are the obstacle avoidance behaviors in different robot control architectures.

V. REVIEW OF ROBOT CONTROL ARCHITECTURES

A. Hierarchical Architectures

Traditionally the architectures were of a hierarchical type, highly influenced by the AI research. This meant a system having an elaborate model of the world, using sensors to update this model, and to draw conclusions based on the updated model. This is often called the sense-plan-act paradigm or deliberative architecture. The algorithms used in this architecture are extremely slow. The disassociation of sensing and acting makes impossible any stimulus-response type of actions (e.g. "a rock is crashing down on me, I should move anywhere"). The fact that the state of the world model may not reflect an actual state of the world makes the architecture unsafe (e.g. the real obstacle may be closer than its model equivalent, leading to a collision).
Another issue that was never resolved in this architecture is handling uncertainty. It may appear in many different forms e.g. as sensor noise, actuator errors or action completion. Lack of feedback makes it uncertain if the robot actually completed a requested action.
The hierarchical architectures did not perform well partly because of the difficulty in modeling of the world, partly because of relying too much on inadequate sensors.
Due to this we will not discuss them any further.

B. Reactive architectures

In 1987 Rodney Brooks [11] revolutionized the field by presenting an architecture based on purely reactive behaviors with little or no knowledge of the world. The reactive architecture is inherently parallel, fast, operates on short time scales and is scalable. There are two general kinds of reactive architectures – subsumption and potential field.
In the subsumption architecture the output is decided by one behavior subsuming the other behaviors. In case of reactive architecture using potential fields for coordination of behaviors, the output emerges as a vector summed from all of the active behaviors.
Reactive architectures eliminate planning and any functions that involve remembering or reasoning about the global state of the robot relative to its environment. That means that a robot cannot plan optimal trajectories (path planning), make maps, monitor its own performance or even select the best behaviors to use to accomplish a task (general planning).

C. Hybrid architectures

The solutions to the drawbacks of reactive architectures appeared in hybrid architectures combining the reactive architectures with modified deliberative architectures.
They combine different representations and time scales, combine closed-loop and open loop execution, may re-use plans and allow dynamic replanning. Hybrid architectures can be loosely subdivided into three categories.
Managerial architectures focus on subdividing the deliberative portion into layers based on the scope of control of each deliberative function.
State hierarchical architectures use the knowledge of the robot's state to distinguish between reactive and deliberative activities. Reactive behaviors are

viewed as having no states, no self-awareness, and function only in the Present. Deliberative functions can be divided into those that require knowledge about the robot's Past state (where it is in a sequence of commands) and about the Future (mission and path planning).

Model-oriented hybrid architectures are characterized by behaviors that have access to portions of the world model, often to the point that they appear to have returned to the Hierarchical monolithic global world model.

VI. OBSTACLE AVOIDANCE ABILITY

A. Reactive architectures

As we have noticed above, the hierarchical architecture does not provide adequate functionality for obstacle avoidance. Much better equipped for this purpose are reactive architectures. The main reason for this is their direct connection of sensors to actuators. Even though the architecture does not have a memory-based world model, but as Brooks said "the world itself is its best model." Continuous sensing guarantees that it is always current, though not always correct, due to the sensors errors or failures.

One of the characteristics of reactive architectures is their ability for graceful degradation of emergent behavior [12]. Let us consider a mobile robot moving to a goal and equipped with a number of different sensors. In the simplest case it will not discriminate between different kinds of obstacles.

Failure of a sensor to detect an obstacle is called false negative. A false positive occurs when a sensor reports a condition that does not exists. From the viewpoint of safety a false negative implies a hazard to be dealt with. One way to mitigate this hazard is to use a set of different sensors invoking suitable behaviors (Fig.1).

For example a robot may use for long distance journey a sonar-avoid behavior. While still some distance off, the robot will turn away from sonar-detected obstacles. But sonar sensors are easily fooled. Smooth surfaces struck at shallow angles reflect sonar beams forward, not back toward the robot – thus no echo is ever detected, causing the sonar system to report a false negative. When this happens, the robot believes that the path is clear, even if it is not.

In this case, the sonar-avoid behavior fails to trigger. But as the robot approaches the acoustically specular object, typically the infrared (IR) detectors will sense an obstacle and drive the robot away. But perhaps along with being too smooth and set at too oblique an angle, the obstacle's surface is also too dark to reflect IR radiation reliably back to the IR obstacle detector. Thus the IR detector may also fail to sense the object, generating a false negative of its own.

With no signal from the IR detector, the IR-avoid behavior does not trigger and the robot collides with the obstacle. The robot bumper now compresses and triggers the Bump-escape behavior. Having failed to avoid the obstacle, the robot must now back up and try to turn away. Typically, the bump sensors can detect, at least crudely, where the collision has occurred, say, on the right, or the left of the robot. This knowledge can give the robot an indication of how to respond. The performance of the robot may suffer, but the robot can still continue its mission.

But suppose there is yet another difficulty, say, that the bumper is not a full-coverage bumper or that it has a dead spot at some point and it is exactly that point that the troublesome smooth, dark oblique object contacts the bumper. Now the bumper fails to report the collision and accordingly, the Bump-escape behavior never triggers. We are left with our robot pressing against the object with all its might.

Fortunately, an over-current sensor is available to detect this condition. When the drive motors are asked to work too hard, as can happen, when they force the robot against an object, motor current goes up. Too high current for too long a time is sensed and is used to trigger a Stall-escape behavior. Although there is no reliable way to tell where the blockage is located with respect to the robot, Stall-escape can at least command the robot to retreat and spin. So, the robot still can move towards its goal.

We have assumed that all the problems are caused by a difficult-to-detect obstacle. But the same behavior would be produced by some inoperative sensors.

Note that the emergent desirable behavior does not require writing a special code that would explicitly instruct the robot to determine if a sensor is working properly. It is just the feature of behavior-based architecture.

The above examples show a crucial difference between the behavior-based approach of reactive architectures and the traditional deliberative approach. In reactive architectures we do not employ a single expensive sensor from which we must demand unattainable levels of precision and reliability. Rather we achieve superior results using a combination of relatively unreliable systems that work together to deliver safe behavior.

In the original subsumption architecture the speed was decided by the behavior subsuming all the other behaviors. Quite often the speed was fixed – the same for all the winning behaviors.

In case of reactive architecture using potential fields for coordination of behaviors [13] the emergent speed is the magnitude of the vector summed from all the active behaviors. It allows therefore to avoid obstacles while moving towards a goal – providing better overall behavior than subsumption architecture.

B. Hybrid architectures

The reactive architectures do not have an ability to monitor performance of the robot or to use e.g. an optimal speed and path to a destination, while avoiding obstacle. This can be mitigated by appropriately designed deliberative layer of a hybrid architecture. For example, in managerial architectures, the deliberative layer may include a Sensing Manager monitoring performance of the robot. If a behavior fails or a perceptual schema detects that sensor values are not consistent or reasonable, the Sensing Manager is alerted. It can then identify alternative perceptual schemas, or even behaviors, to replace the problematic behavior immediately. Imagine a mobile robot in a convoy of robots hauling food to refugees. If the robot had a glitch in a sensor, it should not suddenly stop and think about a problem. Instead it should immediately switch to a back-up plan or even to smoothly slow down while it identifies a back-up plan. Otherwise, the whole convoy would stop, there might be wrecks, etc. The sensing manager working in the background mode, can attempt to diagnose the cause of the problem and correct it.
In some managerial architectures (e.g. SFX [14]) speed control is considered a separate behavior. The safe speed of a robot depends on many influences. If the robot cannot turn in place, it will need to be operating at a slow speed to make the turn without overshooting. Likewise, it may go slower as it goes up or down hills. These influences are derived from sensors, and the action is a template (the robot always slows down on hills), so speed control is a legitimate behavior. But the other behaviors should have some influence on the speed as well. So, these other strategic behaviors contribute a strategic speed to the speed control behavior (Fig.2).
If the strategic speed is less than the safe speed computed from the tactical influences, then the output speed is the strategic speed. But if the tactical safe speed is lower, the output speed is the tactical speed. Tactical behaviors serve as filters on strategic commands to ensure that the robot acts in a safe manner in as close accordance with the strategic intent as possible.

An example of the Model-oriented architectures is the Task Control Architecture (TCA) [15]. TCA has a flavor of an operating system. There are no behaviors per se, however many of low level tasks resemble behaviors. The reactive layer in this architecture called Obstacle Avoidance Layer takes the desired heading and adapts it to the obstacles extracted from the evidence grid virtual sensor. The layer uses a curvature-velocity method (CVM) to factor in not only obstacles but how to respond with a smooth trajectory for the robot's current velocity. Because this architecture does not have direct connection of sensors to actuators, it may have the same problems as the hierarchical architectures of the past – delayed response to events.

CONCLUSION

As can be seen from the above review, robot control architectures are hierarchical or layered. The overall behavior of the architectures emerges from interaction of the participating behaviors. These characteristics justify the system-theoretical approach to evaluation of their safety.
The hierarchical architectures are inherently unsafe. Much safer are the reactive architectures, in which obstacle avoidance is one of the basic and most important behaviors. They lack however an ability to monitor performance of the robot or to use e.g. an optimal speed and path to a destination, while avoiding obstacle. This can be provided by the deliberative layer of the hybrid architectures. From among the different styles of hybrid architectures the most appropriate seems to be the managerial architecture such as e.g. SFX. It should be extended with a communication module implementing an authentication protocol such as the Resurrecting Duckling.

REFERENCES

[1] M. Mataric, Behavior-Based Control: Main Properties and Implications, Proceedings of Workshop on Intelligent Control Systems, International Conference on Robotics and Automation, Nice, France, 1992.
[2] N.G. Leveson, Safeware: System Safety and Computers (Addison Wesley, 1995).
[3] F. Ingrand, R. Chatila, R. Alami, An Architecture for Dependable Autonomous Robots, IEEE/RAS Workshop on Robot Dependability, 2001
[4] N. G. Leveson, Intent Specification: An Approach to Building Human-Centered Specifications, IEEE Trans. on Software Engineering, January 2000.
[5] J.A. Barchanski, Development of safe software for autonomous robots, International Symposium on Robotics and Automation, Queretaro, Mexico, August 2004.
[6] L. Bertalanffy, General Systems Theory: Foundations, Development and Applications,
(G. Braziller, New York, 1969).
[7] I. Asimov, I Robot, (Doubleday, 1950).
[8]N.G. Leveson, A System-Theoretic Approach to Safety in Software-Intensive Systems,

[9] Angie Chandler et al, Digging into Concurrency, Computing Department, Lancaster University, May 2001.
[10] Frank Stajano, Security for Ubiquitous Computing, Wiley, 2002.
[11] R. Brooks, A Robust Layered Control System for a Mobile Robot, IEEE Journal on Robotics and Automation, vol.1, No.1, 1986, pp.1-10.
[12] J.J. Jones, Robot Programming: A Practical Guide to Behavior-Based Robotics, McGraw-Hill, 2004.
[13] R.C. Arkin, Motor Schema Based Navigation foe a Mobile Robot, Proceedings of the IEEE Conference on Robotics and Automation, 1987, Raleigh, N.N. pp.264-71.
[14] R.Murphy, Biological and Cognitive Foundations of Intelligent Sensor Fusion, IEEE Transactions on Systems, Man and Cybernetics, vol.26, no.1, Jan. 1996, pp.156-61.
[15] R. Simmons et al, A Layered Architecture for Office Delivery Robots, Proceedings of Autonomous Agents, 1997, pp.245-52.

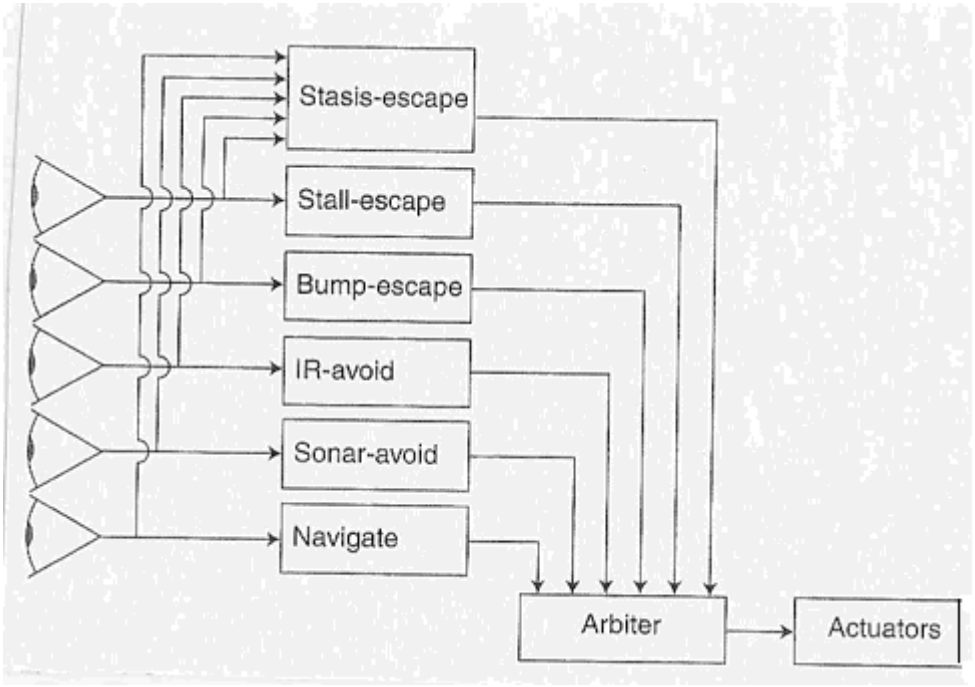

Fig.1. Reactive architecture with graceful degradation.

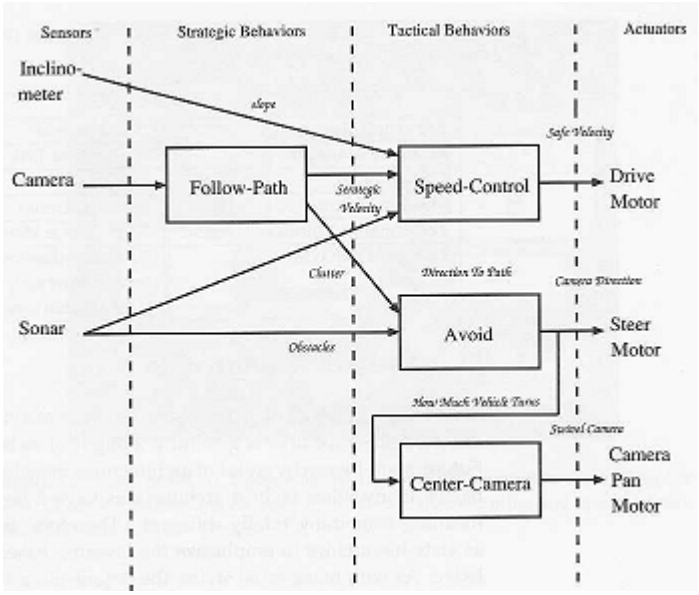

Fig.2. Strategic and tactical behaviors for path following in SFX architecture.